Pedestrian Detection Using Image Blending

Statistics Department

California Polytechnic State University, San Luis Obispo

By Hannah Haggerty

June 2013

Table of Contents

Introduction

After being presented the opportunity to work with cameras of multiple wavelengths, I decided to use one in my senior project. It was brought to my attention that Infrared cameras specifically are commonly used to detect pedestrians. A 2-camera system available to use was able to take visible images and infrared images simultaneously. Another more recent phenomenon is the concept of blending visible images with infrared images. This idea of image blending became a motivating idea that I wanted to somehow incorporate in my senior project. I was inspired to explore the use of blended images for detecting pedestrians using the 2-camera system. The research question I decided to explore was: "Do blended images increase the probability of detecting a pedestrian compared to infrared images alone?"

Background

Pedestrian Detection has become fundamentally more popular and essential over the last decade. Systems are constantly being installed into moving vehicles to help notify drivers of pedestrians. Often vehicles alert the driver if they are a certain distance from an object by cameras that have been mounted into the bumper or some other location on the vehicle. Infrared cameras are the most common form of camera that is used today to detect pedestrians. Infrared cameras appeared beneficial compared to visible cameras because of their ability to better identify objects at night or under darker lighting conditions. These images contain more than the red, green, and blue color scale that visible images are known for, and they offer a substantial amount of clarity when objects are present. For example, in an infrared image it may be easier to differentiate between a pedestrian and a light pole at night compared to a visible image. Even objects that are harder to see in a distance can be more refined in an infrared image than a visible

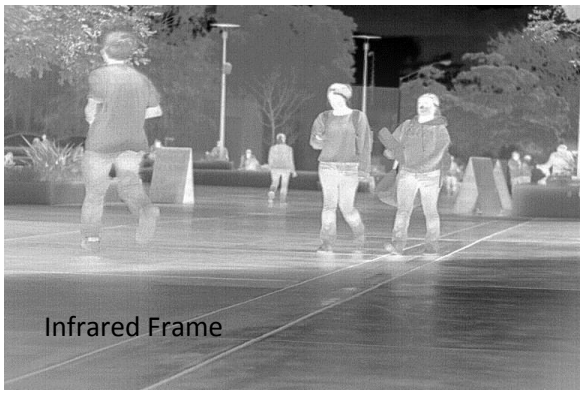
Image 1: Infrared Frame

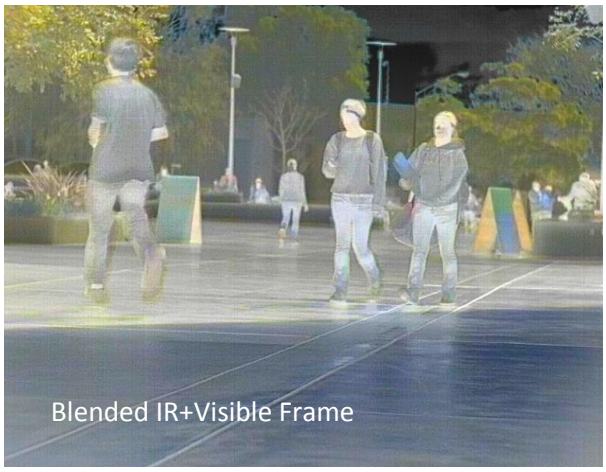
image. But doess an infrared image alone represent the best approach for detecting pedestrians? An infrared image taken with the camera I used for my senior project is displayed to the left. It is apparent to a human observer that pedestrians are present in the image. This exact image was also taken in the original visible frame most of us are used to. The image was taken on the quad on campus at Cal Poly.

Blended Images vs. Infrared Images

Although infrared images seem to do an adequate job of detecting pedestrians, blended images display some similarities to infrared images with some added benefits. Blended images have yet to be investigated when dealing with pedestrian detection, and this is why I felt it would be useful. Image blending can be described as taking two identical frames and merging the pixels from each into one frame. The blended image we used for this project was combining the infrared band with the red, green, and blue visible bands. This combination makes it possible to unite the perks of an infrared image with a visible image, since infrared images may lack some capabilities that a visible image contains. With the blended image we obtained from the quad area it was physically visual that particular areas or objects were better captured than from a single band by itself. The blended images appeared to obtain sections of the frame that can be called “hot spots”. “Hot spots” could be considered areas that illuminate higher temperatures. Sections of the blended image where a pedestrian was present seemed to be warmer than objects such as buildings, trees, structures, etc., at least in the scene of the Quad at Cal Poly. As stated

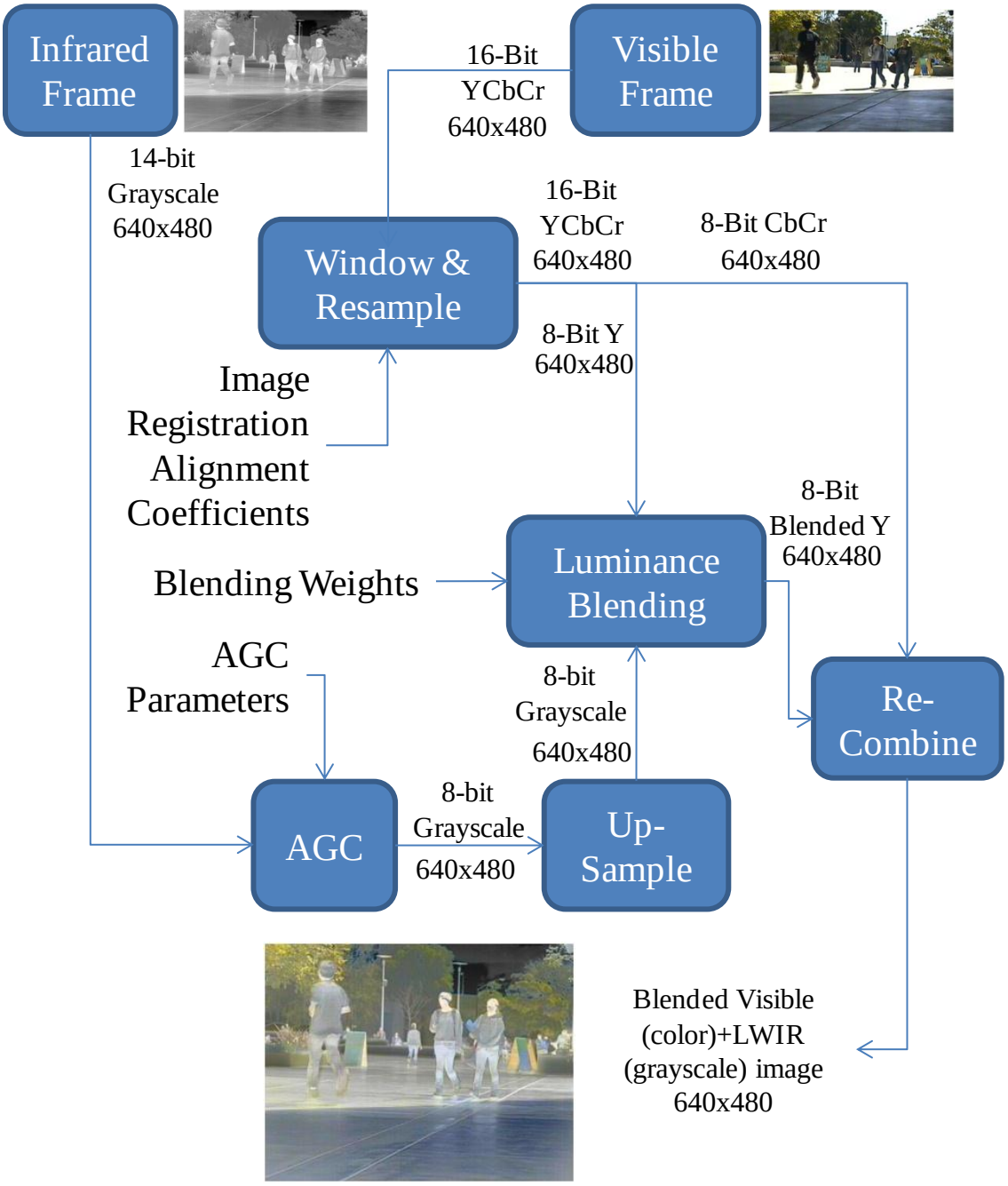
previously, infrared images are quite often used at night where visible images have certain advantages during the day or under lighter lighting conditions. This being the case, the blended should create an increase in the probability of detecting a pedestrian under any lighting condition. The blending of the two frames taken out by the quad can be shown below:





Procedure

The procedure used to blend the images was done using the Mathematical Software MATLAB R2012a. Using MATLAB allowed for determination of pixel locations where objects are identifiable in both frames, infrared and visible. The blending procedure is outlined in the flowchart below.



Once the images were blended the research on determining how to detect pedestrians on a single frame was conducted. One article in particular that I found, nicely displayed the procedures to perform to automatically detect pedestrians in a single frame. My goal then became to implement their methods starting with 2 bands, Infrared band and the Red visible band from the Visible (RGB) frame. This process was also done using MATLAB. Once detecting a pedestrian using

the techniques from the reference with the blended image was accomplished, I researched different probabilities that would be appropriate to apply to this situation.

Pedestrian Detection Algorithm

The main reference used for my project was from Nanda and Davis', "Probabilistic Template Based Pedestrian Detection in Infrared Videos". First, I started by employing the methods in which they use to actually determine if a pedestrian in present in an Infrared frame to my blended image. In collaboration with Gary Hughes, my senior project advisor, we began by masking the blended image. Masking can be described as blacking out exterior objects and environment that is not a pedestrian. Then, highlighting the formation of all pedestrians in white will create a masked image. Statistics such as the mean and standard deviation was calculated for the pedestrian areas (white) and then for the background or non-pedestrians (black). These values were used to find a threshold, the threshold according to Nanda and Davis can be explained by the following equation:

$$\boldsymbol{threshold} = \frac{\sigma 1 \sigma 2}{\sigma 1 + \sigma 2}\, \boldsymbol{ln}\frac{\sigma 1}{\sigma 2} + \frac{\sigma 1 \mu 1 + \sigma 2 \mu 2}{\sigma 1 + \sigma 2}$$

Applying this threshold at each pixel, it is possible to determine whether there is a pedestrian present at that location or not. If the raw masked image at a certain pixel is greater than the threshold at that pixel, then that corresponds to a pedestrian. If the raw masked image at a certain pixel is less than or equal to the threshold value at that pixel, it corresponds to a non-pedestrian.

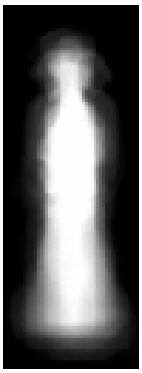
To determine whether the detection was correct or incorrect, Nanda and Davis used this combined probability equation:

$$\boldsymbol{Combined\ Probability(i, j) = \Sigma\ (th(x, y) * p(x, y) + (1 - th(x, y)) * (1 - p(x, y)))}$$

As shown in the equation, the pixel(i, j) calculates the probability of a correct detection if the pedestrian is actually present.

Probabilistic Template

To find the total area of a pedestrian they a probabilistic template as such was used:



Looking at the template, it is apparent the shape resembles a blurred pedestrian. This template was shifted around on the frame and placed on top of each pixel in the image to establish the probability that the pattern is centered on a pedestrian. The combined probability was then calculated at each pixel placement of the probabilistic template.

Bayes' Theorem: Special Case

Since the question I was trying to answer dealt with finding an increase in probability, probability methods were researched to see which would best relate to the situation of using a blended image. It was found that the process Nanda and Davis used above in the Infrared Frame could be classified as a special case of Bayes' Theorem. In a single frame, detecting a pedestrian and not detecting a pedestrian in the image can be considered mutually exclusive and exhaustive events. If A is considered the event that a pedestrian is detected, then the compliment of event A is the event that a pedestrian is not detected. To verify whether the pedestrian was present, given

that the pedestrian was detected could then be found by the conditional probability using based theorem. The conditional probability is calculated as such:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B|A) \cdot P(A) + P(B|A^C) \cdot P(A^C)}$$

Certain portions of this equation are formulas for finding the intersection of the probability of detecting a pedestrian and the pedestrian is present and the intersection of the probability of detecting a pedestrian and the pedestrian is not present.

Pedestrian Detection in an IR Band

Applying this special case of Bayes' Theorem to our infrared band, the events were defined as: A= Pedestrian detected at pixel(x, y), A^c= Pedestrian not detected at pixel(x, y), B= Pedestrian present at (x, y). Using these events, the intersection of events explained previously can be found. Where the intersection of events A and B were considered a Correct Detection rate, the intersection of events A^c and B^c is the Correct Non-Detection, the intersection of A and B^c is type I error, and finally the intersection of A^c and B is type II error. Transmitting this into the conditional probability using the intersections of events we can find the probability of a pedestrian being present given that a pedestrian was detected. This conditional probability is:

$$P(A|B) = \frac{P(A \cap B)}{P(A \cap B) + P(A^C \cap B)}$$

Pedestrian Detection in Two-Bands

The special case of Bayes' Theorem demonstrated above allows for the probability to be found in the situation with a single band, yet it does not directly apply to any more than one band. Although, conditional probabilities and elements of Bayes' theorem can be used in the two-band case. Instead, a different probability model for pedestrian detection using two-bands can be formed. The events would be defined as: A_{IR}= Pedestrian detected in IR frame at pixel(x, y), A_R= Pedestrian detected in R frame at pixel(x, y), and B= Pedestrian present at pixel(x, y). And therefore the probability for a 2 band detections is:

$$P\big((A_{IR}|B) \cup (A_R|B)\big) = P(A_{IR}|B) + P(A_R|B) - P\big((A_{IR}|B) \cap (A_R|B)\big)$$

The elements of this two-band probability model are conditional probabilities such as those using Bayes' Theorem. These elements can be used to find a point estimate for a two-band detection.

Odds Ratio for Two-Band Detection

In addition to being able to find the probability of correctly detecting a pedestrian using the union of the two conditional probabilities for a two-band detection, a benefit of adding another band is can be found by the Odds Ratio:

$$\frac{\{P[(A_{IR}|B) \cup (A_R|B)]\} \cdot \{1 - P(A_{IR}|B)\}}{\{P(A_{IR}|B)\} \cdot \{1 - P[(A_{IR}|B) \cup (A_R|B)]\}}$$

It is known as well, that the distribution of the Log Odds Ratio is approximately normal and therefore sample proportions were able to be assessed from experiments. The sample portions would be of the form shown in this table:

	$(A_{IR}\|B) \cup (A_R\|B)$	$\sim[(A_{IR}\|B) \cup (A_R\|B)]$
$(A_{IR}\|B)$	$\hat{p}_n(1,1)$	$\hat{p}_n(1,0)$
$\sim(A_{IR}\|B)$	$\hat{p}_n(0,1)$	$\hat{p}_n(0,0)$

Using the proportions for each situation we would use them to calculate the sample Log Odds Ratio. The sample Log Odds Ratio and its corresponding Standard Error would be the following:

$$ln\left[\frac{\hat{p}_n(1,1)\cdot\hat{p}_n(0,0)}{\hat{p}_n(1,0)\cdot\hat{p}_n(0,1)}\right] \text{ with Standard Error } \sqrt{\frac{1}{n(1,1)}+\frac{1}{n(1,0)}+\frac{1}{n(0,1)}+\frac{1}{n(0,0)}}$$

Where the value of, n, in the denominators of the equation for standard error is found by dividing the sample proportions by the total number of pixels in the frame(N).

Inference for the Odds Ratio

Another component that can be formulated using this Odds Ratio is a 95% Confidence Interval of the Odds Ratio. To find the Odds Ratio and its 95% confidence interval, estimates from our Nanda and Davis reference were used to calculate the sample proportions found in the table above. The sample proportions for each case were calculated using a truth table. Placing the appropriate probabilities as given in the article into the outcome from each truth table, we were able to find approximations for the sample proportions to use in our formula for the 95% confidence interval of the Odds Ratio. The Odds Ratio cell estimates that we used were:

	$(A_{IR}\|B) \cup (A_R\|B)$	$\sim[(A_{IR}\|B) \cup (A_R\|B)]$
$(A_{IR}\|B)$	.95	.8
$\sim(A_{IR}\|B)$	.05-1.5	.05-1.5

The cases .95 and .8 were those found directly from the articles probabilities. The range of numbers from 0.5 to 1.5 are anticipated values we expect to see for this situation. These anticipated values allowed for us to see how much higher the odds of detecting a pedestrian using a blended image compared to an infrared image would be at the worst case scenario and then at the best case scenario. These estimates and anticipated values were plugged into the (1-α)100% Z-Confidence Interval for the Odds Ratio:

$$e^{\left\{ln\left[\frac{\hat{p}_n(1,1)\cdot\hat{p}_n(0,0)}{\hat{p}_n(1,0)\cdot\hat{p}_n(0,1)}\right]\pm Z_{\frac{\alpha}{2}}\sqrt{\frac{1}{n(1,1)}+\frac{1}{n(1,0)}+\frac{1}{n(0,1)}+\frac{1}{n(0,0)}}\right\}}$$

Summary Findings

Finally, I used Excel to calculate the Log Odds Ratio and 95% Confidence Interval of the Odds Ratio by entering in the estimates and anticipated values of our sample proportions. These results were found for anticipated values of .25 and a total frame size(N) of 1000 pixels:

$$ln\left[\frac{(.95)(.25)}{(.80)(.25)}\right] = 2.498 \quad \text{with a standard error of} \quad \sqrt{\frac{1}{950}+\frac{1}{250}+\frac{1}{250}+\frac{1}{800}} = 0.1015$$

Therefore, the 95% confidence interval for the Odds Ratio is 9.96 to 14.84.

Conclusion

Ultimately, using the estimates from the article and anticipated values we would expect to see for such a situation it was found that the odds of detecting a pedestrian using two-band is higher than the odds of detecting a pedestrian using a single band. As well as, we can say we are 95% confidence, that the benefit is at least a 10 times higher detection rate for two-bands compared to one. Overall, the answer in regards to my research question, "Do blended images increase the probability of detecting a pedestrian compared to infrared images alone?", is yes we do see an increase in this probability.

In the future, I would like to implement these techniques in more than 2-bands to see if there is even more of a substantial increase in the odds or probability. If the odds were already 10 times higher with just the infrared frame blended with the Red frame, I would hope to see that increase per additional band.

Appendix

References:

Nanda and Davis."Probabilistic Template Based Pedestrian Detection in Infrared Videos", 2002. Pg. 1-11.

MATLAB Code:

Image Blending Preparation

```
function img_agc = agc (img, n_rows, n_cols)

    % perform Plateau AGC on unsigned 14-bit image
    max_bit = 2^14;
    hist_bins = zeros (max_bit, 1);
    img_agc = uint8 (zeros (n_rows, n_cols));

    % set up the histogram bins, one for each 16-bit grayscale level
    for i_bin = 1:max_bit
        hist_bins(i_bin) = i_bin - 1;
    end

    % matlab calculates the image histogram
    img_hist = histc (img(:), hist_bins);

    % for Plateau Equalization, clip the bins to some plateau level
    img_hist(img_hist > 150) = 150;

    % calculate the cumulative frequency histogram from the clipped
    % histogram
    cum_hist = cumsum (img_hist);

    % scale the cumulative frequency histogram to 8-bit grayscale
    cum_hist = uint8 (255 * cum_hist / cum_hist(max_bit));

    % use the scaled cumulative frequency histogram as an intensity
    % transform table to map the original 16-bit data to 8-bit level
    for i_row = 1:n_rows
        for j_col = 1:n_cols
            img_agc(i_row,j_col) = cum_hist(img(i_row,j_col) + 1);
        end
    end

return
```

Blend Visible LWIR

```
% read list of equivalent points in the two image frames.  the list of
% points is generated by examining multiple frames of different scenes, and
% determining the pixel locations of objects that are identifiable in both
% image frames.
```

```
% stored in the Excel file as:
%    vis_x(1)  vis_y(1)  lwir_x(1) lwir_y(1)
%    vis_x(2)  vis_y(2)  lwir_x(2) lwir_y(2)
%    ...
%    vis_x(n)  vis_y(n)  lwir_x(n) lwir_y(n)
vis_x = xlsread ('ImageRegistration.xlsx', 'A3:A10');
vis_y = xlsread ('ImageRegistration.xlsx', 'B3:B10');
lwir_x = xlsread ('ImageRegistration.xlsx', 'C3:C10');
lwir_y = xlsread ('ImageRegistration.xlsx', 'D3:D10');

% the list of equivalent points is used to determine a transformation for
% any pixel in the LWIR frame to its equivalent location in the Visible
% frame.  The transformation is:
%    vis_x = a*lwir_x + b*lwir_y + c
%    vis_y = d*lwir_x + e*lwir_y + f
% the coefficients a, b, c, d, e and f are determined by finding the
% best-fit solution to two linear systems, using the list of equivalent
% points.  The linear systems are:
%    [lwir_x lwir_y 1]*[a b c] = vis_x
%    [lwir_x lwir_y 1]*[d e f] = vis_y
% store the coefficients in leading matrix A
n = length (vis_x);
A = ones ([n 3]);
A(:,1) = lwir_x;
A(:,2 )= lwir_y;

% the Matlab \ operator automatically determines the best-fit solution to
% the overdetermined system by least squares.  The coefficients in abc and
% def will provide the best-fit affine transformation, based on the list of
% equivalent points.
abc = A\vis_x;
def = A\vis_y;

% read in the raw image data. for this example, the data are contained in
% video recordings, and only the 20th frame is read in from each file.  the
% cameras used external synchronization so that each frame is taken at the
% same instant in time.

% ir frame from a binary file
nIRrows = 512;
nIRcols = 644;
% visframe 23 = irframe 1
% visframe 920 = irframe 250
irframe = getVideoFrame ('irframe.tif',1);

% visible frame from a bitmap file
 visframe = getVideoFrame ('visframe.tif',1);

% retrieve the visible image dimensions from the data array
[nVISrows nVIScols rgb] = size (visframe);

% upsample the LWIR image to the desired resolution
% first, create a baseline reference frame where integer intersections
% represent the locations of pixels (pixel centers?) in the LWIR frame
[lwirbase_x lwirbase_y] = meshgrid (1:nIRcols, 1:nIRrows);
```

```
% to upsample, create a resampling grid within the LWIR reference frame at
% the desired resolution.  In this case, the desired resolution is the
% native visible frame resolution
desiredres_rows = nVISrows;
desiredres_cols = nVIScols;

% the upsample increment in the LWIR reference frame is the ratio of the
% native LWIR resolution and the desired (upsample) resolution
xIncr = nIRcols/(desiredres_cols + 1);
yIncr = nIRrows/(desiredres_rows + 1);
[lwirres_x lwirres_y] = meshgrid (1:xIncr:nIRcols, 1:yIncr:nIRrows);

% use Matlab's interp2 function to upsample the LWIR image, using the
% LWIR reference grid and the desired resample grid
irframeres = uint16 (interp2 (lwirbase_x, lwirbase_y, double (irframe), ...
                              lwirres_x, lwirres_y, 'bicubic'));

% the resampled image is still 14-bit, so apply AGC to convert to 8-bit
% grayscale.  this step is required to view the image on an 8-bit display,
% and it is also required for the blending operation (later)
irframeresagc = agc (irframeres, desiredres_rows, desiredres_cols);
figure,imagesc (irframeresagc)
axis off
axis equal
colormap('gray')

% the LWIR image has some 'footprint' in the visible image.  need to
% 'extract' that portion of the visible image that lies within the LWIR
% footprint.  the approach here is to re-sample the visible image over a
% grid in the visible frame where each (upsampled) LWIR pixel lies.

% create a baseline reference frame where integer intersections represent
% the locations of pixels (pixel centers?) in the visible frame
[visbase_x visbase_y] = meshgrid (1:nVIScols, 1:nVISrows);

% create a grid that will store the locations in the visible reference
% frame where pixels in the (up-sampled) LWIR image lie.  start with an
% empty array, that is the size of the up-sampled LWIR image, and then
% fill it up pixel-by-pixel, using the [a b c] and [d e f] transformations
visres_x = zeros (desiredres_rows, desiredres_cols);
visres_y = visres_x;

% each point (lwirres_x, lwirres_y) in the up-sampled LWIR image
% corresponds to a location in the visible image, here called (visres_x,
% visres_y).  The point corresponding to (lwirres_x, lwirres_y) in the
% visible reference frame can be found using the [a b c] and [d e f]
% transformations.
for jCol = 1:desiredres_cols
    for iRow = 1:desiredres_rows
        if ((jCol == desiredres_cols) && (iRow == desiredres_rows))
            dog=1;
        end
        visres_x(iRow,jCol) = abc(1)*lwirres_x(iRow, jCol) + ...
                              abc(2)*lwirres_y(iRow, jCol) + abc(3);
```

```
        visres_y(iRow,jCol) = def(1)*lwirres_x(iRow, jCol) + ...
                              def(2)*lwirres_y(iRow, jCol) + def(3);
    end
end

% resample the visible image on the new grid.  the visible image has three
% components, and each component is resampled independently
visframeres = zeros (desiredres_rows, desiredres_cols, 3, 'uint8');
visframeres(:,:,1) = uint8 (interp2 (visbase_x, visbase_y, ...
    double (visframe(:,:,1)), visres_x, visres_y, 'bicubic'));
visframeres(:,:,2) = uint8 (interp2 (visbase_x, visbase_y, ...
    double (visframe(:,:,2)), visres_x, visres_y, 'bicubic'));
visframeres(:,:,3) = uint8 (interp2 (visbase_x, visbase_y, ...
    double (visframe(:,:,3)), visres_x, visres_y, 'bicubic'));

% display the resampled portion of the visible image.
figure,image (visframeres)
axis off
axis equal
colormap('default');

% blending is accomplished by converting the RGB image to YUV format,
% calculating a weighted average of the Y and LWIR frames, then
% re-converting the blendedY+UV back into RGB format.
visprop = 0.1;
irprop = 1 - visprop;
[Y,U,V]=rgb2yuv(visframeres(:,:,1),visframeres(:,:,2),visframeres(:,:,3));
blended = uint8 (visprop*double(Y) + irprop*double(irframeresagc));
rgb_blended = yuv2rgb (blended,U,V,'YUV444_8');

% display the blended image (in color)
figure,image (rgb_blended)
axis off
axis equal

imwrite(rgb_blended,'rgb_blended.jpg','jpg')
imwrite(visframeres,'visframeres.jpg','jpg')
imwrite(irframeresagc,'irframeresagc.jpg','jpg')
```

Get Video Frame

```
%Reads avi, tif, FFF/SEQ and sfmov
function frame = getVideoFrame(videoFileName, frameNr)

%Get file type from video file name
fileType = videoFileName(end-3:end);

%*************************************************************************
%.avi files
%*************************************************************************
if  strcmpi(fileType, '.avi')
    mov = aviread(videoFileName, frameNr);
    [frame, trash] = frame2im(mov);
%     if ndims(frame>2)
```

```
%         img = rgb2gray(frame); %Assumes true color
%     end

%************************************************************************
%.seq files in FFF-format
%************************************************************************
elseif strcmpi(fileType, '.seq')
    [num, text, frame, nrOfFrames] = readfff(videoFileName, frameNr);

%************************************************************************
%.tif files
%************************************************************************
elseif (strcmpi(fileType, '.tif') || strcmpi(fileType, 'tiff'))
    [frame, trash] = imread(videoFileName, frameNr);

%************************************************************************
%.sfmov files
%************************************************************************
elseif strcmpi(fileType, 'fmov') || strcmpi(fileType, 'bmov')
    %Find image size
    fid = fopen(videoFileName);
    while(true)
        line = fgetl(fid);
        if (line == -1)
            break;
        end
        ind = strfind(line, 'XPIXLS');
        if isempty(ind)
            ind = strfind(line, 'xPixls');
        end
        if ~isempty(ind)
            [tmp sizew] = strread(line, '%s %d');
            line = fgetl(fid); %YPIXLS will be in the next line
            [tmp sizeh] = strread(line, '%s %d');
            break;
        end
    end
    fclose(fid);
    %Find start of image data
    fid = fopen(videoFileName);
    while(true)
        line = fgetl(fid);
        if (line == -1)
            break;
        end
        ind = strfind(line, 'DATA');
        if ~isempty(ind)
            break
        end
    end
    status = fseek(fid, (frameNr-1)*(sizeh*sizew)*2, 'cof');
    frame = fread(fid, [sizew sizeh], 'uint16')';
    fclose(fid);
%************************************************************************
%Raw format
%************************************************************************
```

```
elseif strcmpi(fileType, '.raw') || strcmpi(fileType, '.bin')
    [frame, time] = readRaw(videoFileName, frameNr);

%**************************************************************************
%FTS format
%**************************************************************************
elseif strcmpi(fileType, '.fts')
    frame = ftsread2(videoFileName, frameNr);


%**************************************************************************
%unknown file format
%**************************************************************************
else
    error('Unknown file format')
end
```

Convert to an RGB

```
function rgb=yuv2rgb(Y,U,V,yuvformat,convmtrx)
%Converts YUV to RGB
%rgb=yuv2rgb(Y,U,V,yuvformat)
%Version: 4.00, Date: 2007/11/18, author: Nikola Sprljan
%
%Input:
% Y,U,V - Y,U and V components of the frame
% yuvformat - YUV format [optional, default = 'YUV420_8']. See in rgb2yuv.m
%             for supported YUV subsampling formats.
% convmtrx - Conversion matrix [optional, default = 'BT709_1']. The
%            following conversions ase defined. See in rgb2yuv.m
%             for more details.
%
%Output:
% rgb - RGB 3D matrix. rgb(:,:,1), rgb(:,:,2) and rgb(:,:,3) are R, G and
%       B components, respectively.
%
%Uses:
% imresize.m - Matlab Image Processing Toolbox
%
%Note:
% When the input format has the chroma subsampled (e.g. 4:2:0 format),
% upsampling is employed on the chroma components before the conversion
% takes place.
% See in the help of rgb2yuv.m for details on coversion options.
%
% ITU-R BT.601, RGB full range, results in the following transform matrix:
%  1.164   0.000   1.596
%  1.164  -0.392  -0.813
%  1.164   2.017   0.000
% ITU-R BT.601, RGB limited range, results in the following transform matrix:
%  1.000   0.000   1.402
%  1.000  -0.344  -0.714
%  1.000   1.772   0.000
% ITU-R BT.709, RGB limited range, results in the following transform matrix:
%  1.000   0.000   1.570
```

```
%  1.000  -0.187  -0.467
%  1.000   1.856   0.000
%
%Example:
% rgb = yuv2rgb(Y,U,V);

if (nargin < 4)
    yuvformat = 'YUV420_8';
end;
if (nargin < 5)
    convmtrx = 'BT709_l';
end;

if strcmp(convmtrx,'BT601_f')
   load('BT601_f.mat','-mat');
elseif strcmp(convmtrx,'BT601_l')
   load('BT601_l.mat','-mat');
elseif strcmp(convmtrx,'BT601_219')
   load('BT601_219.mat','-mat');
elseif strcmp(convmtrx,'BT709_f')
   load('BT709_f.mat','-mat');
elseif strcmp(convmtrx,'BT709_l')
   load('BT709_l.mat','-mat');
end;

%create the 3D YUV array
yuv = zeros(size(Y,1),size(Y,2),3);
if (strcmp(yuvformat,'YUV420_8'))
    yuv(:,:,1) = double(Y);
    yuv(:,:,2) = imresize(double(U),2,'bicubic');
    yuv(:,:,3) = imresize(double(V),2,'bicubic');
elseif (strcmp(yuvformat,'YUV444_8'))
    yuv(:,:,1) = double(Y);
    yuv(:,:,2) = double(U);
    yuv(:,:,3) = double(V);
end;

%inversion of the transform matrix
T = inv(rgb2yuvT);
rgb = zeros(size(Y,1),size(Y,2),3);
if (yuvoffset(1) ~= 0)
  yuv(:,:,1) = yuv(:,:,1) - yuvoffset(1);
end;
if (yuvoffset(2) ~= 0)
  yuv(:,:,2) = yuv(:,:,2) - yuvoffset(2);
end;
if (yuvoffset(3) ~= 0)
  yuv(:,:,3) = yuv(:,:,3) - yuvoffset(3);
end;
rgb(:,:,1) = T(1,1) * yuv(:,:,1) + T(1,2) * yuv(:,:,2) + T(1,3) * yuv(:,:,3);
rgb(:,:,2) = T(2,1) * yuv(:,:,1) + T(2,2) * yuv(:,:,2) + T(2,3) * yuv(:,:,3);
rgb(:,:,3) = T(3,1) * yuv(:,:,1) + T(3,2) * yuv(:,:,2) + T(3,3) * yuv(:,:,3);
rgb = uint8(round(rgb));
```

Implementing Probabilistic Template

```
% read in infrared image and mask
lwir_image=imread ('lwirframe01.bmp');
lwir_image=lwir_image(:,:,1);
imagesc(lwir_image)
colormap(gray)

% read in the mask
mask=imread ('lwirframe01_mask.bmp');
mask=mask(:,:,1);

%see if the mask works
pedestrians=lwir_image.*mask;
figure,imagesc(pedestrians)
colormap(gray)

% calculate statistics for pedestrians, background
ped_list=double(lwir_image(mask==1));
mean_ped=mean(ped_list);
stdev_ped=std(ped_list);

back_list=double(lwir_image(mask==0));
stdev_back=std(back_list);
mean_back=mean(back_list);

% get probability template
prob_template=imread('prob_template.bmp');
prob_template=double(prob_template(:,:,1));
prob_template=prob_template/255;
```